UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/693,717	10/24/2003	Holly Knight	MS306958.1/MSFTP545USA	1585

27195 7590 05/11/2005

AMIN & TUROCY, LLP
24TH FLOOR, NATIONAL CITY CENTER
1900 EAST NINTH STREET
CLEVELAND, OH 44114

EXAMINER	
CHOW, CHIH CHING	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 05/11/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/693,717	KNIGHT ET AL.	
	Examiner	Art Unit	
	Chih-Ching Chow	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 March 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *24 October 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *8/19/04*, 3/25/05
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to amendment dated March 14, 2005.

2. Per Applicants' request, specification, claims 1, 2, 6, 9-12, and 21-24 have been amended. Claims 1-24 remain pending.

Response to Amendment

3. Applicants' amendment dated 03/14/05, responding to the 11/10/2004 Office action are reviewed respectfully. The Examiner's response are listed as following:

- Objection to Claim 24 for an Informality

  The examiner has reviewed the updated Claim 24 respectfully. The objection to Claim 24 is hereby withdrawn in view of Applicants' amendment to the claim.

- Objection to Claims 9-11 under 35 U.S.C. § 112, First Paragraph

  Examiner's Response: The objection to Claims 9-11 is hereby withdrawn in view of Applicants' amendment to the claims (changed 'constants' to 'constant accessors').

- Objection to Claims 21-24 under 35 U.S.C. § 112, First Paragraph

  Examiner's Response: The objection to Claims 21-24 is hereby withdrawn in view of Applicants' amendment to the claims (changed 'constants' to 'constant accessors').

- Objection to Claim 1 under 35 U.S.C. § 112, Second Paragraph

Examiner's Response: The objection to Claim 1 is hereby withdrawn in view of Applicants' amendment to the claims (clarified 'register component' to 'registry component').

Response to Arguments

4. Applicants' arguments are basically in the following points:

A. Objection to Claims 1 and 12 under 35 U.S.C. § 112, First Paragraph

Examiner's Response: In response to applicant's argument that the 'preference classes' are clearly described in the subject application. Examiner expected that the Applicants can narrow down the scope of 'preference class' even it's described on page 26, such as to further define a set of condition and action classes; however, it's understandable that the 'condition class' and 'action class' can be extended to any possible 'conditions' and 'actions' therefore, it's not possible to define all of them - per Applicants' explanation under REMARKS, stated on page 9, 2nd paragraph, "Conditions or actions are candidates for use by existing applications to be extended such that the application can use the conditions and/or actions. Once an application binds its preference classes to a condition or logic, function candidate conditions or actions simply become conditions or actions." Therefore, the 35 U.S.C. § 112, First Paragraph objection to Claim 1, 12 is hereby withdrawn.

B. Rejection of Claims 1-2, 6-8, and 12-14 Under 35 U.S.C. § 103 (a)

Examiner's Response: The Applicants argued about IBM and Shauhgnessy merely 'pertain to compilation and linking'; 'the subject invention, as claimed,

utilizes executable programs or applications and functions provided herby to enhance other application capabilities, specifically by extending preference classes to allow user of such functions." The Applicants argument is reviewed respectfully, however it is not persuasive for the following reasons:

- Actually, IBM Ada/6000 user can extend its functions by adding a second spec package (that with a second function) into the alib.list file, in fact, the alib.list file functions like the 'EDF', which holds the binding information. The second spec package are the same as the 'second application preference class declarations', which are bound to the function provided by the first application.
- The amended claim 1 recites: "A system for dynamically extending application preference classes comprising:

  a first executable application including one or more functions that are registered in a registry component; and

  an extension component that reads function data from the registry and **binds** a second executable application to the first, wherein second application preference class declarations are bound to the function provided by the first application."; this claim relies on the 'binding' step, without modifying the bindings, the first executable can't combine with the second executable application. - Therefore the prior arts' disclosure does teach what is in the current application.

C. Rejection of Claims 3-5 Under 35 U.S.C. § 103 (a)

Examiner's Response: The Applicants argued about that "As provided in the specification accessors or constant accessors are named groups of objects that provide arguments to conditions and actions in place of a user having to manually specify each such object". The Applicants argument is reviewed respectfully, however it is not persuasive for the following reasons:

- Claim 3 recites: "The system of claim 1, wherein the functions provide conditions."

Claim 4 recites: "The system of claim 1, wherein the functions provide events."

Claim 5 recites: "The system of claim 1, wherein the functions provide accessors."

The spec recites "accessors or constant accessors that provide arguments to conditions and actions" - not **functions**, and it mentions conditions and actions, doesn't mention events.

- Since the claim 1 rejection is maintained (see above), therefore the 35 U.S.C. § 103 rejection to dependent claims 3-5 are also maintained.

D. Rejection of Claims 15-20 Under 35 U.S.C. § 103 (a)

Examiner's Response: The Applicants argued about that "IBM Ada/6000 does not pertain to uninstalling an application. It does not teach notifying dependent applications"; "it teaches displaying an error message to a user when a compiler aborts." The Applicants argument is reviewed respectfully, however it is not persuasive for the following reasons:

- In the specification, 'notify the user' has mentioned many times (see paragraphs 254, 400, 435), it never mentioned 'notifying dependent applications' - the Examiner interprets that the notifying means notifying the user, an 'error message' displayed on the screen is sufficient enough in the sense of notifying the user.

5. Examiner is maintaining the the 35 USC 103 Rejections. For the Applicants' convenience they are listed as following, with the amendments requested by the Applicants. In addition to the original 35 USC § 103 rejection, a 35 USC § 102 rejection is added to the office action.

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless -
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. Claims 1, 9-12, 15 and 21-24 are rejected under 35 U.S.C. 102(b) as being anticipated by Microsoft XP version 5.1.

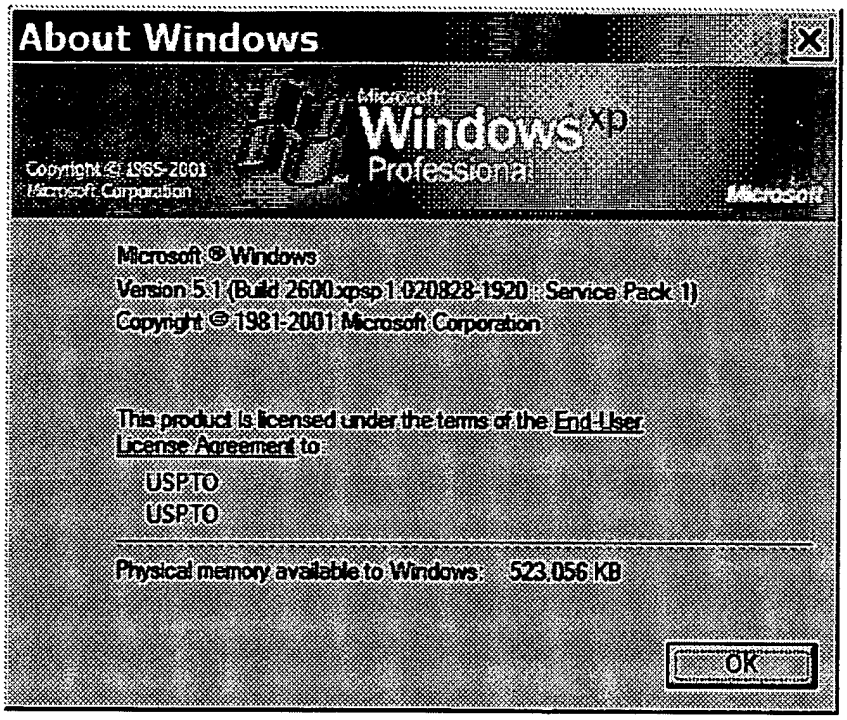
Claim	**Microsof XP**
1. A system for dynamically extending application preference classes comprising: a first executable application including one or more functions that are registered in a registery component; and	Microsoft XP teaches a dynamically extending application preference classes system, which allows user to extend a preference class by simply editing the **extension data file (EDF)**, see the attached printouts below the table.

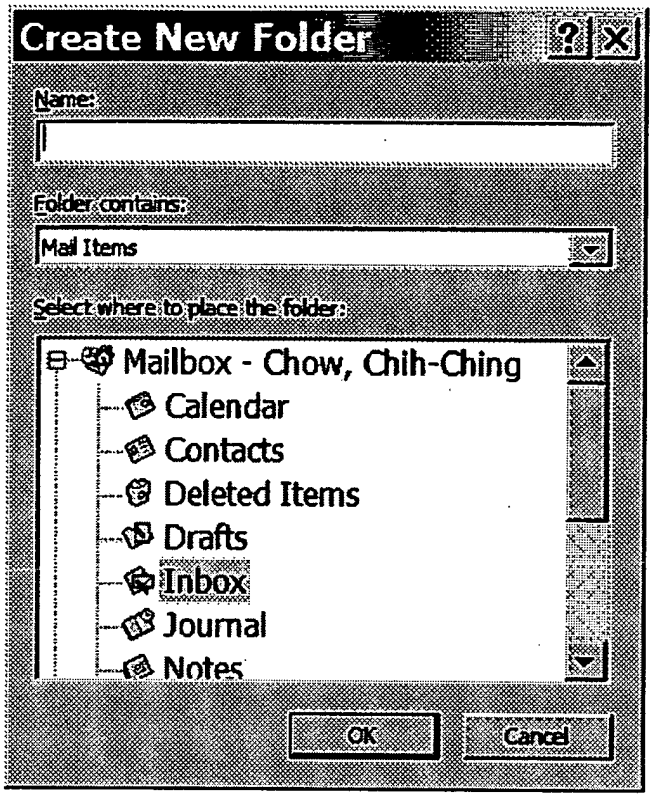
an extension component that reads function data from the registry binds between a second executable application to the first, wherein second application preference class declarations are bound to the functions provided by the first application.	The Microsoft XP version 5.1 allows user to customerize their own PC, for any existing preference class, user can add (extend)/delete(remove)/modify existing registery component (e.g. Personal Folders). The example icon printed below is displayed after user selects "File" → "Folder" → "New Folder", this event allows a user to create a new personal folder for "Mail Items". - a new mail folder name can be a second executable application, which can be **bound with** the original 'Microsoft Outlook Personal Folder'; so the newly added folder name will be displayed when the user clciks 'Personal Folders' when running Microsoft Outlook.
9. A system for extending condition constants comprising: an input component for receiving a constant accessor; and an accessory component for determining the constant accessor value from data stored across a plurality of domains.	Same as claim 1 rejection. Here the input component is the area allows user to enter an input string. The input component receives a 'input constant accessor', which is actually an object accessor to a group of objects in the system domain. There can be more than one constant accessors, it's incremented whenever user addes in a new object.
10. The system of claim 9, wherein the constant accessor is a first order constant accessor.	See claim 1 and 9 rejecitons.
11. The system of claim 9, wherein the constant accessor is an Nth order constant accessor.	See claim 1 and 9 rejecitons.

12. A method for extending application preference class functionality comprising: (a) receiving an extension data file (EDF) containing information about candidate function bindings; (b) registering one or more function bindings in a central data store; and (c) binding a function of a first executable application to a preference class of a second executable application utilizing binding function information located in the central data store.	See claim 1 and 9 rejecitons.
15. A method of uninstalling an application comprising: (a) removing all application registrations from central storage location; (b) removing Program Components; and (c) notifying dependant applications.	For uninstalling, simply do the opposite thing, at "File"→ "Folder" → "Delete" a folder - the designated folder and all associated funtions will be removed.
21. A method for extending constant accessors comprising: receiving a constant accessor; and resolving the value of the constant accessor by searching across application domains.	See claim 1 and 9 rejections.
22. The system of claim 21, wherein the constant accessor is a first order constant accessor.	See claim 1 and 9 rejections.
23. The system of claim 21, wherein the constant accessor is an Nth order	See claim 1 and 9 rejections.

constant accessor.

24. A computer readable medium having stored thereon computer executable instructions for carrying out the method of claim 21.	Any PC with Windows XP installed is a computer readable mediume which can carry out method stated in claim 21.


About Windows
Windows xp
Professional
Copyright © 1985-2001
Microsoft Corporation
Microsoft
Microsoft ® Windows
Version 5.1 (Build 2600.xpsp1.020828-1920 : Service Pack 1)
Copyright © 1981-2001 Microsoft Corporation
This product is licensed under the terms of the End-User License Agreement to:
USPTO
USPTO
Physical memory available to Windows: 523,056 KB
OK



Claim Rejections - 35 USC § 103

8. Claims 1-2, 6-8, 12-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over 'IBM Ada/6000', September 02, 1998; (hereinafter "IBM"); and US Patent No. 6,026,235 by Steven T. Shaughnessy et al. (hereinafter "Shaughnessy"); in view of U.S. Patent No. 6,016,394 by Jeffrey L. Walker (hereinafter "Walker").

Claim	**IBM / Shaughnessy / Walker**
1. A system for dynamically extending application preference classes comprising:	IBM Ada 6000 teaches a compiling environment which allows user to define a first application and further include

a first executable application including one or more functions that are registered in a registery component; and

an extension component that reads function data from the registry binds between a second executable application to the first, wherein second application preference class declarations are bound to the functions provided by the first application.

second application (or more applications). To summarize the IBM Ada / 6000 compiler (refer to 'IBM Ada/6000' Synopsis):

- Ada has packages which are independent computer software components.
- "adalib" is the sublibrary refers to the Ada library which is the current working library
- "alib.list" is crated whenever an 'alibinit' command is entered (a sublibrary 'adalib' is also created). 'alib.list' specifies the current working directory/directories (as long as they are accessible). User can specify program units located in other library/libraries that he/she would need to get the designated program compiled; in other words, user can include other libraries in the current library by editing the alib.list file and **register** all the file objects that he would need in order to get the designated program to be compiled with no error.
- obscure error message would be produced whenever the compiler confronts a library unit is missing during compiling.

For item (a), IBM's 'adalib' functioning as the **instance registry component**. It's also taught by Shaughnessy, see Shaughnessy column 2, lines 12-19, "In

typical operation, a linker receives, either from the user or from an integrated **compiler, a list of object modules desired to be included in the link operation**. (*instance registry component*) The linker scans the object modules from the object and library files specified. After resolving interconnecting references as needed, the linker constructs an **executable** image by organizing the object code from the modules of the program in a format understood by the operating system program loader." In column 3, lines 8-10, "In addition to the additional calls, the approach requires a special **executable** link operation **to bind** in the function entry/exit function calls and the required runtime support for them." For item (b), the application(s) including one or more functions are specified in the **alib.list**.

(c) All the program units been specified in the **alib.list** will be compiled and bound by the compiler. Therefore, the second application can be 'bound' with the first application's functions as long as it has the 'visibility' (accessible) to those functions. Here the '**extension component**' can be the *Ada compiler* itself. It has the visibility to all the files that has been compiled, and can do binding among them.

IBM and Shaughnessy teach the computer feature to include different application components in a computer

compiling/binding environment, however they don't mention 'extending application preference classes' specifically, however, Walker teaches extending application preference for applications in an analogous prior art. In Walker, column 16, lines 50-56, "A prototype smart codes transaction 195 facilitates the **addition** *(extending)* of a new smart code transaction which creates a smart code set and may define **a new smart pick using the smart code set**. By using smart codes, default values may be specified as **preferences** for use in application creation. Smart codes provide a centralized location for **lists of values in a custom target application**." Further more, column 4, lines 59-62, "In the present invention, the concept of "sets" allows an application designer to define all the data on which the target application will possibly operate." (extending *application preferences*).
It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the feature of including multiple application units by extending the application preferences further taught by Walker for the purpose of creating a target software applications for end-users. (see Walker's Abstract line 1).

2. The system of claim 1, wherein the registry component comprises a

For the feature of claim 1 see claim 1 rejection. The *adalib* is the '**instance**

definition registry for storing function definitions and a binding registry for storing binding data.	**registry component**' since adalib contains both 'definition registry' and 'binding registry'; it stores all the function definitions and the binding data.
6. The system of claim 1, wherein functions are only available for binding to specific applications.	For the feature of claim 1 see claim 1 rejection. Functions are only available for binding if they are defined in an application, which are part of the 'to-be-compiled' list (alib.list), see IBM page 6 'Including Other Libraries in Yours'.
7. The system of claim 2, wherein the binding registry receives function binding information from an extension data file (EDF).	For the feature of claim 2 see claim 2 rejection. Again, the 'alib.list' has the same function as the 'extension data file', since it contains all the binding registry information.
8. The system of claim 1, wherein the binding is broken upon removal of the function providing application.	For the feature of claim 1 see claim 1 rejection. An 'obsolete' error will be produced if any program component is missing from the adalib (say **removal** of a function), therefore the 'binding is **broken**'. See IBM page 7, 'Errors During A Compile or Bind'.
12. A method for extending application preference class functionality comprising: (a) receiving an extension data file (EDF) containing information about candidate function bindings; (b) registering one or more function bindings in a central data store; and (c) binding a function of a first	Same as claim 1 rejection. Where alib.list is the **extension data file**; the **central data storage** is the adalib itself; the **binding** is done by the Ada compiler.

application to a second application utilizing binding function information located in central data store.

13. The method of claim 12, further comprising applying acceptance logic to determine whether the second application will accept the binding.	For the feature of claim 12 see claim 12 rejection. See page 10, 2nd paragraph of the Office Action.
14. A computer readable medium having stored thereon computer executable instructions for carrying out the method of claim 12.	The IBM example given in claim 1, an executable file (defaulted **a.out**, see IBM page 5, 1st paragraph) will be created after the **Ada** command is executed (if no compiling and binding error). The Ada compiler contains the **instructions carrying out the method of Ada compiling and binding**, the adalib itself **stores the executable** which is produced after compiling and binding.

9. Claims 3-5 are rejected under 35 U.S.C. 103(a) as being unpatentable over 'IBM Ada/6000', September 02, 1998; (hereinafter "IBM"); and US Patent No. 6,026,235 by Steven T. Shaughnessy et al. (hereinafter "Shaughnessy"); in view of U.S. Patent No. 6,016,394 by Jeffrey L. Walker (hereinafter "Walker"); further in view of 'Compile Time Scheduling of an Ada Subset' by E.W. Giering III et al., Washington Ada Symposium Proceedings, June 1990 (hereinafter "Giering").

Claim	IBM / Shaughnessy / Walker / Giering
3. The system of claim 1, wherein the functions provide conditions.	For the feature of claim 1 see claim 1 rejection. IBM Ada compiler and Shaghnessy teach the feature of functions but does not mention 'conditions' specifically, however Giering teaches this feature in an analogous prior art. In Giering, page 153, section 6.1, 2nd paragraph, '**conditional statements** could be allowed in our model. This would amount to an **implicit conditional compilation** feature'.
4. The system of claim 1, wherein the functions provide events.	For the feature of claim 1 see claim 1 rejection. IBM Ada compiler teaches the feature of functions but does not mention 'events' specifically, however Giering teaches this feature in an analogous prior art. In Giering, page 153, section 6.1, 3rd paragraph, 'Ada **tasks** (*events*) **can also be referenced** by pointer or kept in an array. Making or accepting **an entry call on a task** in such an array is another form of data-dependent execution of a rendezvous primitive'.
5. The system of claim 1, wherein the functions provide accessors.	For the feature of claim 1 see claim 1 rejection. IBM Ada compiler teaches the feature of functions but does not mention 'accessors' specifically, however Giering teaches this feature in an analogous prior art. In Giering, page 143, column2, 3rd paragraph, 'Since any **task** can **call an entry**' (*accessors*) - in Ada, making calls to task is implied.

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the instance registry (compiling list), and IBM's alib.list feature by the including the conditions/ events/ accessors features further taught by Giering for the purpose of scheduling a restricted form when compiling a program (see Giering first paragraph).

10. Claims 15-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over 'IBM Ada/6000', September 02, 1998; (hereinafter "IBM"); and US Patent No. 6,026,235 by Steven T. Shaughnessy et al. (hereinafter "Shaughnessy").

Claim	IBM / Shaughnessy
15. A method of uninstalling an application comprising: (a) removing all application registrations from central storage location; (b) removing Program Components; and (c) notifying dependant applications.	**Uninstalling (removing)** any library unit from alib.list, the adalib will **remove** that specified unit from the working library, and all the dependent units will be recompiled. The **central storage location** is the adalib, it's also an instance registry component (see claim 1 (a) rejection). If any obsolete error has occurred, the **obscure error message** will be displayed on the compiling screen (*notifying dependent applications*). See IBM page 7, 'Errors During A Compile or Bind'.
16. The method of claim 15, wherein the central storage location is an instance registry.	For the feature of claim 15 see claim 15 rejection.

17. The method of claim 16, wherein the instance registry comprises a definition registry and a binding registry.	For the feature of claim 16 see claim 16 rejection. The adalib itself functions the same as '**instance registry**', which contains the definition registry and the binding registry (see claim 1 rejection).
18. The method of claim 17, wherein removing registrations comprises removing registrations in the definition registry and the binding registry.	For the feature of claim 17 see claim 17 rejection. For the rest claim 18 features see claim 15 rejection.
19. The method of claim 15, wherein the notifying dependant applications causes dependant applications to place their dependencies in a NotAvailable state.	For the feature of claim 15 see claim 15 rejection. The 'NotAvailable' is the same as 'obsolete' state, therefore see 'obsolete error message' part of claim 15 rejection.
20. A computer readable medium having stored thereon computer executable instructions for carrying out the method of claim 15.	For the feature of claim 15 see claim 15 rejection. The IBM Ada compiler contains the **instructions carrying out the method of Ada compiling and binding** based on the alib.list, therefore any unit is removed from the alib.list will cause a removal of that unit from the adalib. Again, the adalib itself **stores the executable**, which is produced after compiling and binding.

Conclusion

The following summarizes the status of the claims:

35 USC § 102 rejections: Claims 1, 9-12, 15 and 21-24

35 USC § 103 rejections: Claims 1-8, and 12-20

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:30am - 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306. Any inquiry of a general nature of relating to the status of this application should be directed to the **TC2100 Group receptionist: 571-272-2100.**

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow
Examiner
Art Unit 2192
April 29, 2005

CC

ANTONY NGUYEN-BA
PRIMARY EXAMINER